A software radio platform based on ARM and FPGA

Xin.Yang [1], Zhengxiang.Li[2]

[1]*Embedded system design, Communication University of China, Beijing, China*
[2]*Embedded system design, Communication University of China, Beijing, China*

Abstract. The rapid rise in computational performance offered by computer systems has greatly increased the number of practical software radio applications. A scheme presented in this paper is a software radio platform based on ARM and FPGA. FPGA works as the coprocessor together with the ARM, which serves as the core processor. ARM is used for digital signal processing and real-time data transmission, and FPGA is used for synchronous timing control and serial-parallel conversion. A SPI driver for real-time data transmission between ARM and FPGA under ARM-Linux system is provided. By adopting modular design, the software radio platform is capable of implementing wireless communication functions and satisfies the requirements of real-time signal processing platform for high security and broad applicability.

1 Introduction

With the rapid development of Software Radio technology, Software Radio technology has become a hotspot of research. Software Radio is a new architecture for wireless communication based on digital signal processing and microelectronics technology. In this paper, an ARM and FPGA based software radio platform is designed according to the needs of actual system in digital signal processing and wireless communication, which take full advantage of the flexibility of ARM and the parallelism of FPGA.[1] The choice of ARM is Freescale’s IMX6, which has a rich instruction set and programming flexibility; FPGA has a great advantage in the speed and parallel computing, suitable for requirements for real-time data transmission.

This paper begins with an overview of the software radio platform, including hardware components and data-stream transmission on the platform. It then moves on to describe the design of data transmission interface between ARM and FPGA. An explanation of how the SPI driver can implement real-time data transmission between ARM and FPGA under ARM-Linux system will be presented after that. The sequential logic and the transmission of multi-data in FPGA is explained at the end of this paper.

2 Platform Overview

The core architecture of this platform is based on ARM system and FPGA system. ARM system uses a Freescale’s IMX6Q microprocessor which encompasses a quad-core platform running up to 1.2 GHz. FPGA system uses a chip of Altera's Cyclone III series chips which contains six groups of phase-locked loop CLK management and can run up to220MHz. A/D converter module uses ADS7886 which is a 12-bit and 1MHz Sample Rate Serial ADC. The serial interface in ADS7886 is controlled by the CS and SCLK signals for connections with microprocessors.

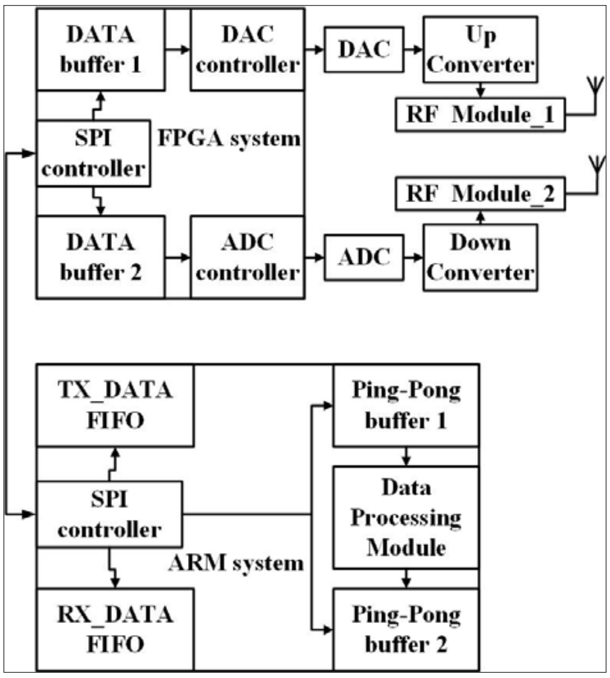

Figure 1. Diagram of embedded software radio platform.

Figure.1 shows the diagram of embedded software radio platform. The platform is mainly composed of ARM system, FPGA system, ADC&DAC module and

RF module. The function of ARM system is storing the real-time data and processing the real-data. The ARM chip IMX6Q complete instructions and data transmission by the address of read and write bus.

Embedded Linux kernel is responsible for mission management and integration of enhanced SPI serial data transmission, guaranteeing real-time data transmission. The role of FPGA is synchronous timing controlling and serial-parallel converter. FPGA is in charge of data acquisition from the ADC, digital-to-analogue conversion by controlling the DAC and controlling the speed of data transmitting to ARM.[2] ADC&DAC module consists of ADC, DAC and peripheral circuit. RF module is used to receive and process radio-frequency signal.

2.1 Data Transmission Design

2.1.1 Real-time Data Transmission Design

The difficulty of platform design lies in the design of real-time data transmission interface, because it is the data transmission channel, and is also a bridge to connect the system front-end and back-end.[3] Compared with general processing platform, the real-time platform requires demands on faster response time, and deviation of synchronous sequential logic will cause serious consequences. It means that the hardware of platform must have the ability to process and transmit data with high speed.[4]

What is innovative about real-time data transmission interface of this platform is that FPGA and ARM use master-slave mode control structure by Enhanced serial peripheral interface of ARM. ARM SPI bus controller works on slave mode, and FPGA SPI bus controller works on mater mode to provide chip select signal and CLK signal for ARM. FPGA can read valid data processed by ARM from data cache of ARM through SPI bus, and then FPGA convert serial data read out from ARM into parallel data. After that, FPGA will send parallelized data to DAC which will handle the digital to analog conversion. Meanwhile, FPGA can write avail data sampled by ADC into data cache of ARM through SPI bus, and then ARM will store valid data and process valid data.

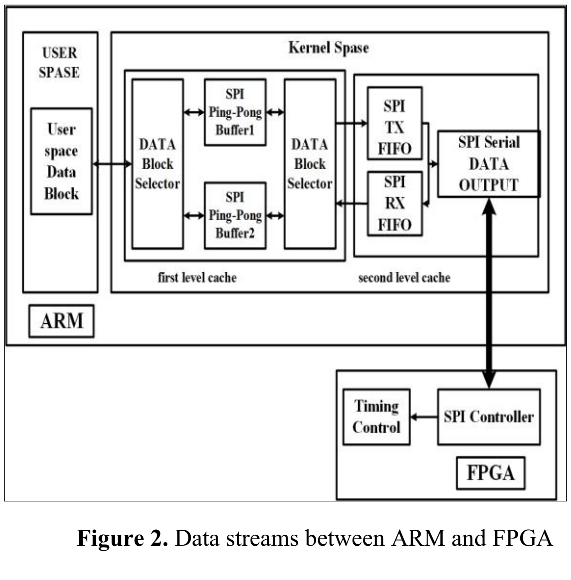

Figure 2. Data streams between ARM and FPGA

Figure.2 shows data streams between ARM and FPGA. By reason that the Enhanced serial peripheral interface is a full-duplex, synchronous, four-wire communication block. Once ARM-Linux user space has completed their data handling, ARM-Linux kernel space can receive processed data in blocks from ARM-Linux user space (the maximum data block size is 120Kbytes), and then FPGA can read out serial processed data from ARM and convert serial data into parallel data. After that, FPGA will send parallelized data to DAC, which will handle the digital to analog conversion. Meanwhile, ARM-Linux kernel space can transfer data sampled by ADC in blocks to ARM-Linux user space (the maximum data block size is 120Kbytes), ARM can process data in ARM-Linux user space. Real-time system must have ability to process and transmit data with high speed. In order to ensure constant and continuous data transmission, ARM-Linux system uses ping-pong operation of two pieces of data buffer (the maximum data buffer size is 120K bytes) to achieve real-time data in series. When the valid data block of user space is writing into Ping-pong Buffer0 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer1 which has completed writing into the valid data block. Similarly, when the valid data block of user space is writing into Ping-pong Buffer1 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer0 which has completed writing into the valid data block. In this way, SPI TX_FIFO can read valid data from data buffer continuously. (first level cache of ARM internal data)

FPGA SPI bus controller provide chip select signal and clk signal for ARM to control the read or write speed of RAM. Therefore, FPGA can read processed data from data buffer of ARM at a constant rate, FPGA also can write real-time data sampled by ADC into data buffer of ARM at a constant rate.

As ARM SPI bus controller works on slave mode, and FPGA provides chip select signal and CLK signal for ARM. In order to ensure ARM can output data continuously at a constant rate, half- interruption method is used to collect and transfer data.[5] When the number of valid data is less than 32, ARM-Linux will generate an interrupt and use interrupt service routine to fill SPI TX_FIFO. (second level cache of ARM internal data)

In the design of data transmission interface, ARM is responsible for the data caching and processing, FPGA is responsible for timing control for ARM.

2.1.2 FPGA Data Transformation Design

In the platform, FPGA is responsible for completing serial-parallel conversion and synchronous timing control of ARM as well as ADC & DAC. FPGA works as the coprocessor together with the ARM, which serves as the core processor. ARM is used for digital signal processing and real-time data transmission, and FPGA is used for synchronous timing control and serial-parallel conversion.

FPGA works as the coprocessor together with the ARM, which serves as the core processor. ARM is used for digital signal processing and real-time data transmission, and FPGA is used for synchronous timing control and serial-parallel conversion. A SPI driver for real-time data transmission between ARM and FPGA under ARM-Linux system is provided.[6] The design of real-time data transmission interface takes full advantage of ARM and FPGA. ARM contains various peripheral interfaces and enough storing resource. Besides, it can run an embedded operating system Linux to manage all the tasks. FPGA has excellent reconfiguration flexibility and parallel computational ability, which enhance the performance of both control and computation of the platform. The CLK of ADC & DAC and timing control of ARM is provided by FPGA. ARM is used to process data and store data.

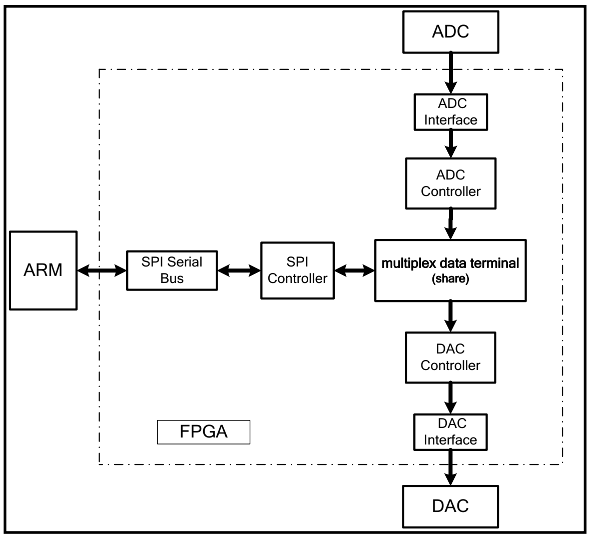

Figure 3. FPGA SPI bus controller

Figure.3 shows SPI bus data streams and SPI bus controller in FPGA.FPGA SPI bus controller provide chip select signal and clk signal for ARM to control the read or write speed of RAM.A SPI driver for real-time data transmission between ARM and FPGA under ARM-Linux system is provided.data streams between ARM and FPGA. By reason that the Enhanced serial peripheral interface is a full-duplex, synchronous, four-wire communication block. Once ARM-Linux user space has completed their data handling, ARM-Linux kernel space can receive processed data in blocks from ARM-Linux user space (the maximum data block size is 120Kbytes), and then FPGA can read out serial processed data from ARM and convert serial data into parallel data. After that, FPGA will send parallelized data to DAC.

Therefore, FPGA can read processed data from data buffer of ARM at a constant rate, FPGA also can write real-time data sampled by ADC into data buffer of ARM at a constant rate.A/D converter module uses ADS7886 which is a 12-bit and 1MHz Sample Rate Serial ADC. The serial interface in ADS7886 is controlled by the CS and SCLK signals for connections with microprocessors.

Multiplex data terminal is a module inside FPGA, which can implement sharing data channel between ARM and ADC&DAC. FPGA can read processed data from data buffer of ARM through the SPI bus. Meanwhile, FPGA can write real-time data sampled by ADC into data buffer of ARM through the SPI bus.The data transmission between ARM and FPGA is a full-duplex transmission.

2.2 ARM-Linux SPI bus driver design

Driver is the interface between the operating system and hardware and it shielded the hardware details of the application. Only the hardware system combined with the highly efficient and reliable driver could it work properly in the operating system.

In order to realize the communication between ARM and FPGA in ARM-Linux system for master-slave mode to transmit data. A specialized SPI bus driver and software interface is designed to provide data transmission channel between the operating system and hardware.

Once ARM-Linux user space has completed their data handling, ARM-Linux kernel space can receive processed data in blocks from ARM-Linux user space (the maximum data block size is 120Kbytes), and then FPGA can read out serial processed data from ARM and convert serial data into parallel data. After that, FPGA will send parallelized data to DAC, which will handle the digital to analog conversion. Meanwhile, ARM-Linux kernel space can transfer data sampled by ADC in blocks to ARM-Linux user space (the maximum data block size is 120Kbytes), ARM can process data in ARM-Linux user space.[7] Real-time system must have ability to process and transmit data with high speed. In order to ensure constant and continuous data transmission, ARM-Linux system uses ping-pong operation of two pieces of data buffer (the maximum data buffer size is 120K bytes) to achieve real-time data in series. When the valid data block of user space is writing into Ping-pong Buffer0 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer1 which has completed writing into the valid data block. Similarly, when the valid data block of user space is writing into Ping-pong Buffer1 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer0 which has completed writing into the valid data block. In this way, SPI TX_FIFO can read valid data from data buffer continuously.

Figure.4 shows SPI bus data streams and FIFO in ARM.SPI TX_FIFO can read valid data from Ping-pong Buffer1 which has completed writing into the valid data block. Similarly, when the valid data block of user space is writing into Ping-pong Buffer1 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer0 which has completed writing into the valid data block. In this way, SPI TX_FIFO can read valid data from data buffer continuously.

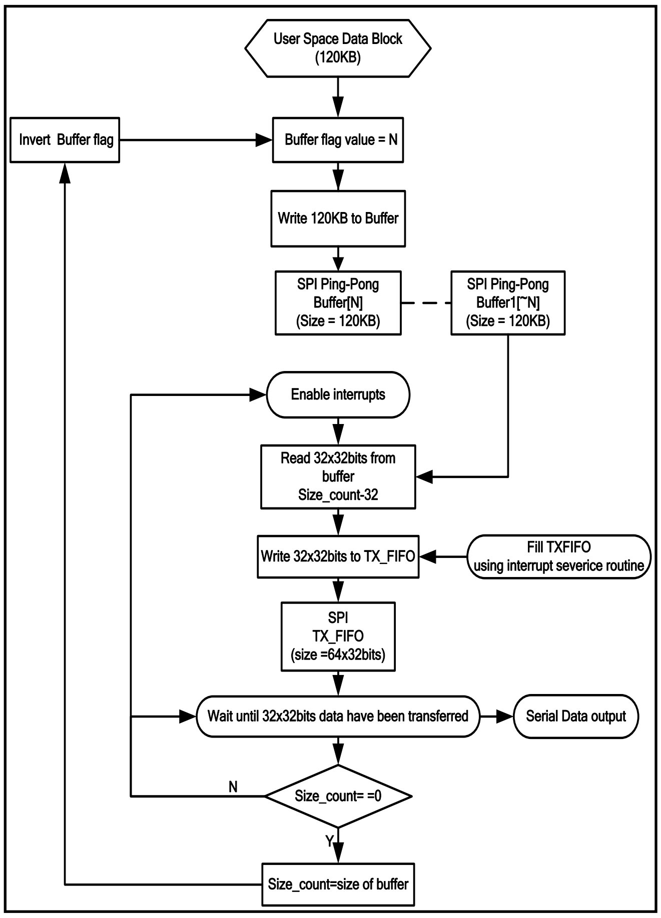
User Space Data Block (120KB)
Invert Buffer flag
Buffer flag value = N
Write 120KB to Buffer
SPI Ping-Pong Buffer[N] (Size = 120KB)
SPI Ping-Pong Buffer1[~N] (Size = 120KB)
Enable interrupts
Read 32x32bits from buffer Size_count-32
Write 32x32bits to TX_FIFO
Fill TXFIFO using interrupt severice routine
SPI TX_FIFO (size =64x32bits)
Wait until 32x32bits data have been transferred
Serial Data output
N
Size_count= =0
Y
Size_count=size of buffer

Figure 4. Software flow pattern of SPI bus driver

The processed data in ARM-Linux user space can be written into ARM-Linux kernel space in blocks, and then FPGA can read out serial processed data from ARM through the SPI bus. Meanwhile, the real-time data sampled by ADC can be transmitted to ARM-Linux kernel space through the SPI bus, and then the real-time data in ARM-Linux kernel space can be written into ARM-Linux user space in blocks.

In order to ensure constant and continuous data transmission, ARM-Linux system uses ping-pong operation of two pieces of data buffer (the maximum data buffer size is 120K bytes) to guarantee the real-time data processing. When the valid data block of user space is writing into Ping-pong Buffer0 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer1 which has completed writing into the valid data block. Similarly, When the valid data block of user space is writing into Ping-pong Buffer1 in kernel space, SPI TX_FIFO can read valid data from Ping-pong Buffer0 which has completed writing into the valid data block. The time that FPGA finish read data from a data buffer is about , this time is longer than the time that Linux controls worst response time (50ms).[8] In this way, FPGA can transmit data continuously.

3 Conclusion

This software radio platform takes full advantage of the flexibility of ARM and the parallelism of FPGA, realizes the design of high-speed data transmission and processing.

With the rapid development of Software Radio technology, Software Radio technology has become a hotspot of research. Software Radio is a new architecture for wireless communication based on digital signal processing and microelectronics technology.[9] In this paper, an ARM and FPGA based software radio platform is designed according to the needs of actual system in digital signal processing and wireless communication, which take full advantage of the flexibility of ARM and the parallelism of FPGA. The choice of ARM is Freescale’s IMX6, which has a rich instruction set and programming flexibility; FPGA has a great advantage in the speed and parallel computing, suitable for requirements for real-time data transmission.

This work presented an embedded scalable cluster platform with software defined radio applications. FPGA serves as the coprocessor and works together with the ARM which serves as the core processor in this platform. ARM is used to complete digital signal processing and real-time data transmission, And FPGA is used to complete synchronous timing control and serial-parallel conversion.[10] A SPI driver for real-time data transmission between ARM and FPGA under ARM-Linux system is provided.

Experimental results show that the real-time and high speed performance of the software radio platform can meet the design requirements, and it is capable of implementing wireless communication functions and satisfies the requirements of real-time signal processing platform for high security and broad applicability.

References

1. Wang Ya-ting, *High-speed data acquisition technology research based on ARM and FPGA*, 2007.
2. Long Zai-chuan, Feng Lin. "*Embedded CCD acquisition system based on ARM and FPGA*", Optoelectronics • Laser, vol. 18, no.11, pp.1296-1298, 2007.
3. ZHU Xiao-peng, XIAO Tie-jun, Zhao Hui. "*ARM+FPGA for realtime data acquisition system design*", Computer Engineering and Design, vol. 30, no.13, pp.3088-3090, 2009.
4. XU Xin-shun, JIA Zhi-ping. Embedded Linux Application Programming, People's Posts & Telecom Press, 2008.
5. J. van de Belt, P. D. Sutton, and L. E. Doyle, "*Accelerating software radio: Iris on the Zynq SoC*" in Very Large Scale Integration (VLSISoC), 2013 IFIP/IEEE 21st International Conference, pp. 294-295, 2013.
6. Zhao Xiao-pu, Li Li, Song Hua "*Gigabit Ethernet interface in S3C2440A System*", Electronic Design and Application, vol. 8, pp.101-103, 2007.
7. A. Love, W. Zha, and P. Athanas. "*In pursuit of instant gratification for FPGA design*" in Field

Programmable Logic and Applications (FPL), 2013 23rd International Conference , pp. 1-8, 2013.

8. J. van de Belt, P. D. Sutton, and L. E. Doyle, "*Accelerating software radio: Iris on the Zynq SoC*" in Very Large Scale Integration (VLSISoC), 2013 IFIP/IEEE 21st International Conference, pp. 294-295, 2013.
9. J. Delorme, J. Martin, A. Nafkha, C. Moy, F. Clermidy, P. Leray, and J. Palicot. "*A FPGA partial reconfiguration design approach for cognitive radio based on NoC architecture*" in Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008.
10. Wu Wei, Hao Jian-Xin, Lu Li. "*High-speed data acquisition system based on ARM and FPGA*", Modern electronic technology, no.1, pp.126-128, 2009.